Organic Form Language
Language Reference Manual

Eric Larson
el2264
March 1, 2006
COMS W4115

1. Language Overview

Organic Form Language, OFL, is based on L-systems. OFL adds structure through blocks and methods. Code blocks translate into scoping directives. Methods are named code blocks. Indentation is used to define code blocks.

2. Lexical Conventions

There are six types of tokens. They are keywords, identifiers, comments, operators, new line followed by white spaces, and integers. Operators, white spaces, and newlines are used to separate tokens.

2.1. Keywords

The keywords begin with an upper case ASCII letter, which is followed by only upper and lower ASCII letters.

The keywords are:

Right, Left, Draw, Thickness, length, Angle, Color, Generations, Start, Final, PerGen.

2.2.identifiers

Identifiers begin with a lower-case ASCII letter, and may have only upper and lower-case ASCII letters, integers, and underscores, '_'.

2.3.Comments

Comments begin with a pound-sign, '#', and continue until the first newline.

2.4. Integers

Integers are one or more decimal digits between '0' and '9'.

2.5. White Spaces following a Newline

A white space immediately after a newline indicates a code block. Otherwise white spaces are ignored.

2.6. Operators

Non-letter ASCII symbols are used as operators. They are one character long.
They are:

':','=','-','+'

3. Semantics

3.1 Indentation

Like Python, indentation defines code blocks. Only code blocks are indented. Initialization of environment members,3.2.1.2, and Generations,3.2.2.2, always begin on a new line and are not indented. Any method declaration will not be indented, and will immediately precede one code block.

Indentation expresses scoping within code blocks.
Each instruction inherits the environment from the last instruction that is more shallowly indented. The execution environment is not effected by those instructions more deeply indented. Only an instruction which is more shallowly indented than the immediately subsequent instruction will have any effect on the execution environment of another instruction.

For example, here are a series of commands in OFL. Let underscore, '_', represent white space.

__a1
__b2
____c3
__d4

In this case, a1 will not effect c3 because b2 is not more deeply indented. The only execution environment being modified is c1 because it will inherit b1's environment, and any changes to the environment caused by b1.

3.2. Keywords

A keyword is a directive, environment member, or a flow control statement.

3.2.1Directives and Environment Members

Both directive and environment members change the runtime environment of an OFL program, but directives also direct the display program.

3.2.1.1.Directives

Directives either rotate the direction in which future drawing is done, or direct the display program to draw.

They do not take a value, so they only can share a line with comments and white spaces.

3.2.1.1.1Turning Commands
Right and Left rotates the direction of drawing clockwise or counter clockwise by the value of Angle, 3.2.1.2.4.
3.2.1.1.1.1.Right
Turns direction of drawing clockwise.
3.2.1.1.1.2. Left
Turns direction of drawing counter-clockwise.
3.2.1.1.1.3. Draw
Directs the display program to create a visible mark of Length, 3.2.1.2.2, long from the current position in the direction of drawing with a width of Thickness,3.2.1.2.1, pixels in of the Color,3.2.1.2.3 . This is the only means in OFL to paint on the screen.

3.2.1.2. Environment Members
Environment members only change the state of the runtime of an OFL program.

They always take an integer value, so
they are always followed by an equals sign, '=', and either an integer or a plus or minus sign, '-', '+'.

3.2.1.2.1. Thickness
The value of Thickness dictates the thickness of lines painted when Draw is called.
3.2.1.2.2. Length
The value of length dictates the length of a line painted when Draw is called.
3.2.1.2.3. Color
The value of Color dictates the color of a line painted when Draw is called.
3.2.1.2.4 Angle
The value of Angle dictates how much direction of drawing is changed when Right or Left is called.

3.2.2.Flow Control Statements
Flow control is based around the concept of a generation in recursive calls. A generation is a unit of growth of the drawn form. Methods may be declared to be “per generation”,

which tells the system that they may only be invoked once for each generation in each call path.

For example, if “twig” is a per generation method and recursively calls itself twice, and is called by “branch” twice, twig will execute twice during the first generation, then four times the second generation, then eight times during the third generation. If twig is not per generation, then it will run indefinitely.

3.2.2.1. PerGen

PerGen is a modifer in the declaration of a method indicating that the method should only be invoked once in any call path per generation.

This never takes a value, but is only used as a modifier for a method declaration, so it only appears after white space following an method identifier, but before the method identifier's colon. For example, “<\n>my_method PerGen:<\n>”

3.2.2.2.Generation

Sets the total number of times PerGen methods will be invoked in each call path. This always takes an integer value, so it is followed by an equals sign '=' and an integer. It is only set once in any program, and it is always on its own line with no preceding spaces but may have trailing white spaces and comments before the subsequent newline.

3.2.2.3. Start

The value of Start is the first method call made during the start of any run of the OFL program.

This always takes an identifier value, so it is followed by an equals sign ':' and an identifier. It is only set once in any program, and it is always on its own line with no preceding spaces but may have trailing white spaces and comments before the subsequent newline.

3.2.2.4. Final

Final is an optional control statement in a method. It is the alternative value of the method during the final generation. It is similar to an identifier definition in that it has a declaration line followed by a code block which is more deeply indented than the declaration. It

is different in that the declaration line is indented. The Final statement begins with the string “Final” followed immediately by a colon. Except for a comment and trailing spaces, the declaration line must otherwise be empty. The Final definition ends at the first line following the declaration with indentation equal or more shallow than that of the declaration line.

3.3. Identifiers

An identifier names a user defined method.

3.3.1.Identifier Definition

The Identifier definition is an identifier declaration line followed by a code block.

3.3.2.Identifier Declaration

An identifier declaration names the method and includes any modifiers. The identifier must be unique.
The identifier is followed by a modifier and colon, or just a colon. Following the colon could be either a comment or a newline.

3.3.3. Identifier Method Block

A method block contains only indented lines. Only method blocks contain indented lines.

3.4.Operators

3.4.1.Assignment Operators

The assignment operators give a value to a keyword or identifier, which are the “lvalue”.
An equals sign is used if, and only if, an integer value being set.

3.4.1.1. Colon

Colons set the lvalue to code block.
A colon may possibly have a modifier between it and the lvalue.
A colon is the end of a statement on a line, so it is only followed by white space or a comment before the newline.

3.4.1.2 Numeric Assignment

Numerical assignment maybe abbreviated by using an increment or decrement value instead of an integer.

3.4.1.2.1 Equals

An equals sign is required to set an lvalue to an integer. It sits between an lvalue, and either an integer value or a minus or plus sign.

3.4.1.2.2 Minus

Minus may be used instead of a integer in a numeric assignment to indicate decrement by one the value of lvalue.

3.4.1.2.3 Plus

Plus may be used instead of a integer in a numeric assignment to indicate increment by one the value of lvalue.

3.4.2 Pound

Pound indicates the beginning of a comment. Everything between a pound sign and the first subsequent newline is a comment and is ignored. A comment may appear after any statement.